Lecture 3

10-701 Spring 2019

These materials are a more or less a complete summary: more details will be discussed in class and in the assigned readings.

1 Announcements

- Please check Piazza for the new submission rules. You cannot submit handwritten assignments. If you have completed HW1 already and handwritten it you have until 5pm on 1/23 to submit it.
- You have only 30 attempts to submit your code on Autolab. Do not use Autolab to debug your code...

2 Classification

We will talk today about classification. We have sets of points with features $\mathbf{x}_k$ that are paired with labels y_k for $k = 1....n$. Based on the existing data, we want to predict the label y_t for a new sample t with features $\mathbf{x}_t$. Let's consider here binary classification, where $y \in \{0, 1\}$.

2.1 Decision Rules

We want to give an optimal decision for the new label y given $\mathbf{x}$. This will involve determining the joint probability $p(\mathbf{x}, y)$, and then making a decision. The difficult part is to estimate the joint distribution, as we will discuss later.

2.1.1 Bayes decision rule

We write the posterior probability of the label given the data:

$$P(y = i|\mathbf{x}) = \frac{P(\mathbf{x}|y = i)P(y = i)}{P(x)} = \frac{P(\mathbf{x}|y = i)P(y = i)}{\sum_j P(\mathbf{x}|y = j)P(y = j)} = q_i(\mathbf{x})$$

Note: this is not a "Bayesian" method.

Bayes Test: We pick the label that has the maximum posterior probability:
If $q_0 > q_1$ then we choose 0, otherwise 1.
We can also compare $P(\mathbf{x}|y = 0)P(y = 0)$ and $P(\mathbf{x}|y = 1)P(y = 1)$ without computing the normalizing factor.

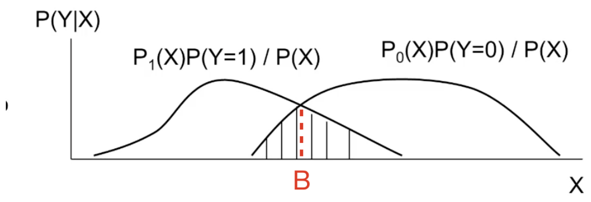
Bayes Error: What is the probability of error?
First we estimate the risk:

$$r(x) = min(q_0(\mathbf{x}), q_1(\mathbf{x}))$$

This is the probability of making a mistake because we always pick the label i with the highest $q_i(\mathbf{x})$. However, there is still a probability that we are wrong: this probability is the probability of the less likely label.

The Bayes Error is our expected risk over $\mathbf{x}$

$$\begin{aligned}\text{Bayes Error} &= E[r(x)] \\ &= \int min(q_0(\mathbf{x}), q_1(\mathbf{x}))p(x)dx \\ &= \int_{-\infty}^{B} P(\mathbf{x}|y=1)P(y=1) + \int_{B}^{\infty} P(\mathbf{x}|y=0)P(y=0) \\ &= P(y=1)\int_{-\infty}^{B} P(\mathbf{x}|y=1) + P(y=0)\int_{B}^{\infty} P(\mathbf{x}|y=0)\end{aligned}$$



The Bayes Error is the lower limit of the error that you can get with any classifier. A classifier that achieves this error rate is an optimal classifier.

However, his is hard in practice because:

- Hard to estimate conditional / joint distribution of $\mathbf{x}$ and y from finite data.
- The definition of the risk above is using a 0-1 loss which is discontinuous and not convex and is hard to optimize over.

0-1 loss:

$$L(\hat{y}, y) = I(\hat{y} \neq y) = \begin{cases} 0 & \text{if} \quad y = \hat{y} \\ 1 & \text{if} \quad y \neq \hat{y} \end{cases}$$

Certain classifiers make assumptions to learn $P(\mathbf{x}|y)$ more easily (e.g. Naive Bayes assumes that the individual features x_j are independent given y). Another strategy is to use a surrogate loss function (e.g. hinge loss), which is usually a convex function that is easier to optimize over which is used as a proxy. More on that later. You might also care about some mistakes more than others and chose to optimize another definition of risk that weights the mistakes differently.

See **[CB] 1.5** and **[KM] Chapter 1**.

Types of Classifier

- Generative: learn the distribution of the data, e.g. Naive Bayes.
- Discriminative: learn a decision boundary directly, e.g. Decision Tree.
- Instance based: use observations directly, such as KNN.

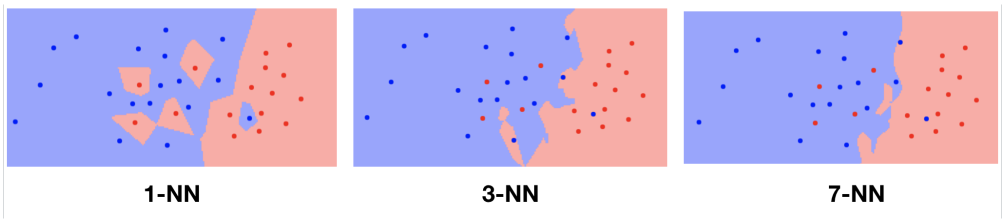
3 K-Nearest Neighbors (KNN)

For a good overview of the KNN algorithm, check out this chapter from Hal Daumé III's Book http://ciml.info/dl/v0_99/ciml-v0_99-ch03.pdf. You can also check out this chapter for a discussion of training and testing: http://ciml.info/dl/v0_99/ciml-v0_99-ch02.pdf.

As we said in class, the biggest sin in Machine Learning is to use your test data in training or allow information about your test data to infiltrate your training.

Do not use your test data during training!

- Additionally, we discussed in class what happens when we chose K=1 (in training) and in testing. Choosing K=1 creates (in testing) a Voronoi tesselation of the points. See this beautiful video for a manual construction of a Voronoi diagram: https://www.youtube.com/watch?v=yDMtGT0b_kg
- We also discussed how the training and test error change as a function of K.
- In KNN multiple settings of the algorithm have to be determined, such as K, the metric used or whether to attribute weights to the individual points.
- Typically, one uses the training set to determine the hyper-parameters for a classifier.
- We saw in class that using the optimal value for K in training (which is 1), would not necessarily lead to good test performance. Indeed we discussed the bias-variance trade-off and saw how it's applied in the KNN setting: at very low K, there is a lot of variance. At very high K, we tend towards always predicting the majority class, and therefore we have less variance and more bias.
- We saw how to use cross-validation on the training set to determine the hyper-parameter K. We discussed multiple types of cross-validation (N-fold CV and leave-one-out CV).
- We discussed overfitting in the case of KNN. When K becomes smaller, the model complexity increases, and the model becomes more prone to overfitting. We have more smooth regions for high K and more granular regions for low K.
- You can use the following applet to play with different settings of K, and other settings: http://vision.stanford.edu/teaching/cs231n-demos/knn



- We saw that KNN can be also expressed probabilistically, and we can show that it's trying to approximate the bayes decision rule. Let's take a point $\mathbf{x}$ and take a fixed K. Now consider the volume V of the sphere v_x that contains the K nearest neighbors of $\mathbf{x}$. Also *assume that* $\mathbf{x}$ *is uniformly distributed in* v_x. Then:
  - Because we assumed $\mathbf{x}$ is uniformly distributed, The probability of points falling in v_x is $p(\mathbf{x})V$. Given our data, our estimate of this probability is K/N, which is the proportion of training points that fall within v_x. Therefore:

$$p(\mathbf{x})V = \frac{K}{N}$$
$$p(\mathbf{x}) = \frac{K}{NV}$$

  - Similarly

$$p(\mathbf{x}|y)V = \frac{K_i}{N_i}$$
$$p(\mathbf{x}) = \frac{K_i}{N_iV}$$

- Our estimate of $P(y=i)$ is $\frac{N_i}{N}$, the proportion of points with label i.
- Now we can compute our estimate of q_i:

$$\begin{aligned} q_i(\mathbf{x}) &= \frac{P(\mathbf{x}|y=i)P(y=i)}{P(x)} \\ &= \frac{\frac{K_i}{N_i V}\frac{N_i}{N}}{\frac{K}{NV}} \\ &= \frac{K_i}{K}. \end{aligned}$$

  When classifying $\mathbf{x}$ we look at the K neighbors and we choose the label with the most counts in K. This is the same as choosing i with the maximum $q_i(\mathbf{x})$ in this case since $q_i(\mathbf{x}) = \frac{K_i}{K}$. Therefore, under this specific set of assumptions, we can see that KNN is following the bayes decision rule.
- In practice, these assumptions are not always true. Otherwise, KNN would always be optimal.